# 情報理工学部 SN コース 2 回
# セキュリティ・ネットワーク学実験 2
# 課題 6 レポート

2600200443-6
Yamashita Kyohei
山下 恭平

December 21 2021

# 1　概要

TCP を用いて、接続許可リストにあるクライアントのみに対し、サーバが保有しているファイル一覧を送信し、クライアントから指定されたファイルを送信するプログラムを作成する。ネットワークセキュリティの向上を意図して、許可された IP アドレスを持つクライアントだけが、通信を行うように実装した。

# 2　外部仕様

## 2.1　サーバ側

サーバは起動と同時にクライアントからの接続を待機する。許可されたクラインアントから接続が行われた時、図 1 のように接続先の IP アドレスを表示し、送信したファイル名を表示し、通信を終了する。

```
kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP % ./6_server
conected :127.0.0.1
data.pngを送信
kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP %
```

図 1　許可した相手との通信

許可されていない相手から通信が行われた場合は図 2 のように、許可されていない趣旨を表示し、通信を終了する。

```
kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP % ./6_server
許可していないIPアドレスからのアクセス
kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP %
```

図 2　許可していない相手との通信

## 2.2　クライアント側

クライアントは起動時、図 3 のように、第二引数にホスト名、第三引数にポート番号を入力する必要がある。

指定した相手と接続を行うとき、初めに入力したホスト名の IP アドレスを表示し、サーバにあるダウンロード可能ファイル一覧を表示する。ファイル一覧が表示された後、ダウンロードしたいファイル番号をキーボードから入力することでダンロードを行うことができる。図 4 はダウンロードまでの様子を示したものである。

もし、自信の IP アドレスがサーバに許可されていない時、”connection refused” と表示する。

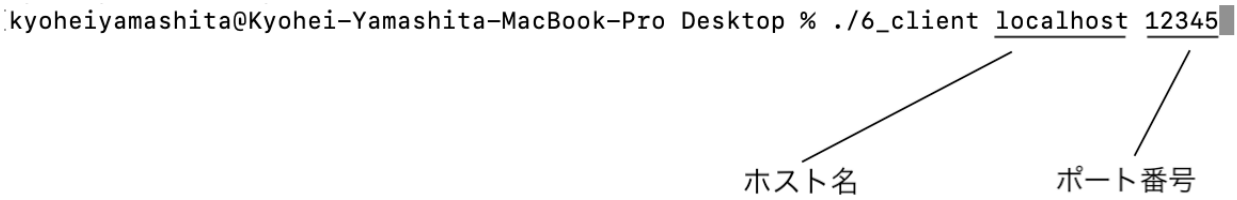


図 3　クライアント起動時

```
kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro Desktop % ./6_client localhost 12345
ip address : 127.0.0.1
Downloadable files
[0] 3-1_client   [1] 3-1_client.c   [2] 3-1_server
[3] 3-1_server.c   [4] 3-2_client   [5] 3-2_client.c
[6] 3-2_server   [7] 3-2_server.c   [8] 3-3_client
[9] 3-3_client.c   [10] 3-3_server   [11] 3-3_server.c
[12] 3-4_client   [13] 3-4_client.c   [14] 3-4_server
[15] 3-4_server.c   [16] 3-5_client   [17] 3-5_client.c
[18] 3-5_server   [19] 3-5_server.c   [20] 3-6_client
[21] 3-6_client.c   [22] 3-6_server   [23] 3-6_server.c
[24] 5-1_client   [25] 5-1_client.c   [26] 5-1_server
[27] 5-1_server.c   [28] 5-2_client   [29] 5-2_client.c
[30] 5-2_server   [31] 5-2_server.c   [32] 5-3_client
[33] 5-3_client.c   [34] 5-3_server   [35] 5-3_server.c
[36] 6_client     [37] 6_client.c   [38] 6_server
[39] 6_server.c   [40] acceptlist.txt   [41] data.png
filenumber >41
download data.png%
```

図 4　サーバと通信を行う時

```
kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro Desktop % ./6_client localhost 12345
ip address : 127.0.0.1
Downloadable files
connection refused
```

図 5　IP アドレスが許可されていないとき

# 3　内部使用

## 3.1　サーバ側

サーバ側ではファイル名を取得するときに ls コマンドを使用している。ls コマンドを仕様するにあたって、サーバがあるディレクトリを自身で設定する必要がある。この処理を「char *cmd」に書き込むことで行っている。クライアントと接続後、「acceptlist.txt」から許可された IP アドレスを読み込み、クライアントの IP アドレスと比較を行い、一致するものがない場合、クライアントに「refuse」と送り通信を終了する。IP アドレスが許可されているとき、ls コマンドを実行し、ファイル名一覧をクライアントへ送信する。クライアント

から送り返されたファイル名のファイルを再びクライアントへ書き込むことで、通信を終了する。

### 3.2 クライアント側

クライアント側ではサーバに接続後、サーバからファイル名一覧が送信されてくるので、それを配列 filenames に格納する、その後、ダウンロードしたいファイルを指定しファイル名をサーバへ送り返し、送られてきたファイルを read することでファイルを取得している。もし、サーバから「refuse」の文字列が送られてきた時、IP アドレスが許可されていないので、その趣旨を示すメッセージを出力し通信を終了する。

## 4 実行例

今回の実行では以下の内容を「acceptlist.txt」に書き込んだ。この時、自身の IP アドレスは「192.168.11.14」である。

Listing 1 acceptlist.txt

```
1    192.168.11.14
```

今回は自分 IP アドレスからの接続は許可されているので、サーバからファイル一覧が送られており、指定したファイルをダウンロードすることができている。実行の様子を図 6 に示す。

```
kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP % ./6_server
conected :192.168.11.14
data.pngを送信
kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP %
```

```
kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro Desktop % ./6_client 192.168.11.14 12345
ip address : 192.168.11.14
Downloadable files
[0] 3-1_client   [1] 3-1_client.c   [2] 3-1_server
[3] 3-1_server.c   [4] 3-2_client   [5] 3-2_client.c
[6] 3-2_server   [7] 3-2_server.c   [8] 3-3_client
[9] 3-3_client.c   [10] 3-3_server   [11] 3-3_server.c
[12] 3-4_client   [13] 3-4_client.c   [14] 3-4_server
[15] 3-4_server.c   [16] 3-5_client   [17] 3-5_client.c
[18] 3-5_server   [19] 3-5_server.c   [20] 3-6_client
[21] 3-6_client.c   [22] 3-6_server   [23] 3-6_server.c
[24] 5-1_client   [25] 5-1_client.c   [26] 5-1_server
[27] 5-1_server.c   [28] 5-2_client   [29] 5-2_client.c
[30] 5-2_server   [31] 5-2_server.c   [32] 5-3_client
[33] 5-3_client.c   [34] 5-3_server   [35] 5-3_server.c
[36] 6_client   [37] 6_client.c   [38] 6_server
[39] 6_server.c   [40] acceptlist.txt   [41] data.png
filenumber >41
download data.png%
kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro Desktop % ls
6_client
6_client.c
data.png
```

図 6 IP アドレスが許可されている時

しかし、localhoat(127.0.0.1) からの接続は許可されていないので、サーバからファイル一覧が送られてくることはなく、警告文を表示して即座に通信を終了している。実行の様子を図 7 に示す。

```
kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP % ./6_server
許可していないIPアドレスからのアクセス
kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro TCP %
```

```
kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro Desktop % ./6_client localhost 12345
ip address : 127.0.0.1
Downloadable files
connection refused
kyoheiyamashita@Kyohei-Yamashita-MacBook-Pro Desktop %
```

図 7 IP アドレスが許可されていないとき

# 5　ソースコード

以下はソースコードである、コードの説明はコード内のコメントアウトにて行っている。

## 5.1　サーバ側

Listing 2　server

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdlib.h>
4  #include <sys/types.h>
5  #include <sys/socket.h>
6  #include <netinet/in.h>
7  #include <arpa/inet.h>
8  #include <pthread.h>
9  #include <errno.h>
10 #include <string.h>
11 #include <fcntl.h>
12
13 //fgets()の改行文字を削除するための関数
14 void lntrim(char *str)
15 {
16     char *p;
17     p = strchr(str, '\n');
18     if (p != NULL)
19     {
20         *p = '\0';
21     }
22 }
23
24 int main(int argc, char *argv[])
25 {
26     int sock0;
27     struct sockaddr_in addr;
28     struct sockaddr_in client;
29     socklen_t len;
30     int sock;
31     int errocheck;
32     int fd,ret,n;
33     char buf[2048];
34     char str[256];
35
36     //サーバがあるディレクトリのパスを入力
37     char *cmd = "/bin/ls /Users/kyoheiyamashita/VSCode/NW実験/TCP";
38
```

```
39      //許可するIPアドレス一覧表
40      char fname[] = "acceptlist.txt";
41
42      FILE *fp; // FILE型構造体
43      int refusecheck = 0;
44
45      /* ソケットの作成 */
46      sock0 = socket(AF_INET, SOCK_STREAM, 0);
47      if (sock0 < 0)
48      {
49          perror("socket");
50          printf("%d\n", errno);
51          return 1;
52      }
53
54      /* ソケットの設定 */
55      addr.sin_family = AF_INET;
56      addr.sin_port = htons(12345);
57      addr.sin_addr.s_addr = INADDR_ANY;
58      errocheck = bind(sock0, (struct sockaddr *)&addr, sizeof(addr));
59      if (errocheck < 0)
60      {
61          perror("bind");
62          printf("%d\n", errno);
63          return 1;
64      }
65
66      /* TCPクライアントからの接続要求を待てる状態にする */
67      errocheck = listen(sock0, 5);
68      if (errocheck < 0)
69      {
70          perror("listen");
71          printf("%d\n", errno);
72          return 1;
73      }
74
75      /* TCPクライアントからの接続要求を受け付ける */
76      len = sizeof(client);
77
78      sock = accept(sock0, (struct sockaddr *)&client, &len);
79      if (sock < 0)
80      {
81          perror("accept");
82          printf("%d\n", errno);
83          return 1;
84      }
85
```

```
86      fp = fopen(fname, "r"); // ファイルを開く。失敗するとNULLを返す。
87      if (fp == NULL)
88      {
89          printf("%s file not open!\n", fname);
90          return -1;
91      }
92
93      //テキストファイルからIPアドレスを読み込み//
94      while (fgets(str, 256, fp) != NULL)
95      {
96          int l = 0;
97
98          while (1)
99          {
100             if (str[l] == '\n')
101             {
102                 str[l] = '\0';
103                 break;
104             }
105
106             l++;
107         }
108
109         //テキスト内に一致するIPアドレスがあればそのIPアドレスを表示//
110         if (strcmp(str, inet_ntoa(client.sin_addr)) == 0)
111         {
112             printf("conected :%s\n", str);
113
114             refusecheck = 1;
115
116             fclose(fp);
117
118             break;
119         }
120     }
121
122     //一致するIPアドレスがない時//
123     if (refusecheck == 0)
124     {
125         puts("許可していないIPアドレスからのアクセス");
126
127         strcpy(buf,"refuse");
128
129         //クライアントへ「refuse」と送信//
130         write(sock, buf, sizeof(buf));
131
132         //通信の終了//
```

```
133         close(sock);
134         close(sock0);
135         return 0;
136     }
137 
138     //lsコマンドでディレクトリ内のファイル名を取得//
139     if ((fp = popen(cmd, "r")) != NULL)
140     {
141         while (fgets(buf, sizeof(buf), fp) != NULL)
142         {
143             lntrim(buf);
144 
145             //ファイル名をクライアントへ送信
146             write(sock, buf, sizeof(buf));
147         }
148 
149         pclose(fp);
150     }
151 
152     //クライアント側readを止めるための文字列を送信//
153     errocheck = write(sock, "HELLO", 5);
154     if (errocheck < 0)
155     {
156         perror("write");
157         printf("%d\n", errno);
158         return 1;
159     }
160 
161     //クライアントから送信するファイル名を受信//
162     read(sock, buf, sizeof(buf));
163 
164     printf("%sを送信\n",buf);
165 
166     //指定されたファイルをオープン//
167     fd = open(buf, O_RDONLY);
168     if (fd < 0)
169     {
170         close(sock);
171 
172         close(sock0);
173 
174         return 0;
175     }
176 
177     //ファイルをクライアントへ送信//
178     while ((n = read(fd, buf, sizeof(buf))) > 0)
179     {
```

```
180         ret = write(sock, buf, n);
181         if (ret < 1)
182         {
183             perror("write");
184             break;
185         }
186     }
187 
188     /* TCPセッションの終了 */
189     close(sock);
190 
191     /* listen するsocketの終了 */
192     close(sock0);
193 
194     return 0;
195 }
```

## 5.2 クライアント側

Listing 3 client

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <sys/types.h>
6  #include <sys/socket.h>
7  #include <netinet/in.h>
8  #include <arpa/inet.h>
9  #include <errno.h>
10 #include <netdb.h>
11 #include <errno.h>
12 #include <fcntl.h>
13 
14 int main(int argc, char *argv[])
15 {
16     struct sockaddr_in server;
17     struct addrinfo hints, *res;
18     struct in_addr addr;
19     int sock;
20     char buf[2048];
21     char filenames[1024][2048];
22     int n;
23     int errocheck;
24     int portnum;
25     int fd;
26     int filecount = 0;
```

```
27      int filenumber;
28      char ipadd[16];
29
30      //ホスト名をIPアドレスへ変換
31      portnum = atoi(argv[2]);
32
33      memset(&hints, 0, sizeof(hints));
34      hints.ai_socktype = SOCK_STREAM;
35      hints.ai_family = AF_INET;
36      if ((errocheck = getaddrinfo(argv[1], NULL, &hints, &res)) != 0)
37      {
38          printf("error %d\n", errocheck);
39          return 1;
40      }
41
42      addr.s_addr = ((struct sockaddr_in *)(res->ai_addr))->sin_addr.s_addr;
43      inet_ntop(AF_INET, &addr, ipadd, sizeof(ipadd));
44      printf("ip address : %s\n", ipadd);
45
46      /* ソケットの作成 */
47      sock = socket(AF_INET, SOCK_STREAM, 0);
48      if (sock < 0)
49      {
50          perror("socket");
51          printf("%d\n", errno);
52          return 1;
53      }
54
55      /* 接続先指定用構造体の準備 */
56      server.sin_family = AF_INET;
57      server.sin_port = htons(portnum);
58
59      /* 127.0.0.1はlocalhost */
60      inet_pton(AF_INET, ipadd, &server.sin_addr.s_addr);
61
62      /* サーバに接続 */
63      errocheck = connect(sock, (struct sockaddr *)&server, sizeof(server));
64      if (errocheck < 0)
65      {
66          perror("connect");
67          printf("%d\n", errno);
68          return 1;
69      }
70
71      memset(buf, 0, sizeof(buf));
72
73      puts("Downloadable files");
```

```
74
75      //"HELLO"が送られてくるまでサーバからファイル名を受信する
76      while ((n = read(sock, buf, sizeof(buf))) > 0)
77      {
78
79          if (strcmp(buf, "HELLO") == 0)
80          {
81              break;
82          }
83
84          //IPアドレスが許可されていない場合"refuzed"が送られてくる。
85          if (strcmp(buf, "refuse") == 0)
86          {
87              puts("connection refused");
88
89              close(sock);
90
91              return 0;
92          }
93
94          printf("[%d] %-10s ", filecount, buf);
95
96          //受信したファイル名を配列へ格納
97          strcpy(filenames[filecount], buf);
98
99          filecount++;
100
101         if ((filecount % 3) == 0)
102         {
103             putchar('\n');
104         }
105
106         memset(buf, 0, sizeof(buf));
107     }
108
109     //受信したいファイルを選択//
110     printf("filenumber >");
111
112     scanf("%d", &filenumber);
113
114     printf("download %s", filenames[filenumber]);
115
116     //受信したいファイル名をサーバへ送信//
117     write(sock, filenames[filenumber], sizeof(filenames[filenumber]));
118
119     //ファイル書き込み用の空のファイルを生成//
120     fd = open(filenames[filenumber], O_WRONLY | O_CREAT | O_EXCL, 0600);
```

```
121     if (fd < 0)
122     {
123         puts("すでに同じ名前のファイルが存在");
124 
125         write(sock, "erro", sizeof("erro"));
126 
127         close(sock);
128 
129         return 1;
130     }
131 
132     //サーバから送られてきたデータの読み込みと書き込み//
133     while ((n = read(sock, buf, sizeof(buf))) > 0)
134     {
135         errocheck = write(fd, buf, n);
136 
137         if (errocheck < 1)
138         {
139             perror("write");
140             break;
141         }
142     }
143 
144     /* socketの終了 */
145     close(sock);
146 
147     return 0;
148 }
```